## REMARKS/ARGUMENTS

Claims 1, 9, and 17 are pending in the present application. Claims 2-8, 10-16, and 18-24 are canceled. By this response, claims 1, 9, and 17 are amended to address the rejection of claims 1, 9, and 17 under 35 U.S.C. § 112, second paragraph, by replacing the phrase "a large random integer less than the large prime number minus one" with the phrase "the primitive root raised to a power of a large random integer less than the large prime number minus one" to correct an error in the claims, as disclosed in the specification and indicated by the Examiner. Applicants respectfully request that the Examiner enter these amendments to place the claims in better condition for appeal. Reconsideration of the claims is respectfully requested.

### I. 35 U.S.C. § 112, Second Paragraph

The Examiner rejects claims 1, 9, and 17 under 35 U.S.C. § 112, second paragraph, as being indefinite for failing to particularly point out and distinctly claim the subject matter, which applicants regard as the invention. This rejection is respectfully traversed.

The Examiner states:

> The claims disclose that a first set of values includes ("a client-generated random value, a large prime number, a primitive root of the large prime number and a large random integer less than the large prime number minus one"). However, the specification (pg. 11) and the drawings (Fig. 7A, 706), disclose a client generated random value (rc) a large prime number (p) a primitive root (g) and the primitive root *raised to the power of a random integer less than the large prime number minus one.*
>
> It is unclear whether the Applicant intends on claiming sending the large integer x, or the calculated value g^x in the claim language.

Final Office Action dated August 16, 2007, pages 2-3.

Claims 1, 9, and 17 are amended by replacing the phrase "a large random integer less than the large prime number minus one" with the phrase "the primitive root raised to a power of a large random integer less than the large prime number minus one" to correct the error in the claims, as disclosed in the specification and indicated by the Examiner. Therefore, Applicants respectfully request that these after final amendments are entered and the rejection of claims 1, 9, and 17 under 35 U.S.C. § 112, second paragraph, is withdrawn.

### II. 35 U.S.C. § 103, Obviousness

The Examiner rejects claims 1, 9, and 17 under 35 U.S.C. § 103 as being unpatentable over *Peyravian et al.*, "Method for Protecting Password Transmission", Computers and Security, Vol. 9, No. 5,

pp.466-469, 2000, hereinafter referred to as *Peyravian*, in view of *Trostle* (U.S. Patent Number 6,718,467). This rejection is respectfully traversed.

With respect to independent claims 1, 9, and 17, the Examiner states:

Regarding Claim 1,

> Peyvarian teaches the computer network, comprising: a client and a server connected by a network connection,
>
> wherein the client has a userid and a password associated with the client *("The user submits his userid (id) and password (pw) to the client" pg. 4);*
>
> wherein the client requests access to the server by sending a first set of values to the server *("The client generates a random value (rc) and sends id and rc to the server" pg.4)* ;
>
> wherein the server responds to the client by generating a first random value and sending the token to the client; *("The server generates a random value (rs) and sends it to the client" pg. 4). The Examiner interprets the nonce (rs) as the token.*
>
> the client changes the password by computing a hash of the userid and a new password to form a new digest *(idpw_digest_new = Hash(id, new_pw), pg. 6, Peyravian),*
>
> creating a mask *(auth_token_mask, pg. 7, Peyravian),* computing a message authentication code, and by exclusive-oring the mask with the new digest to form a result *(protected_idpw_new = protected_idpw_new XOR auth_token_mask, pg. 7, Peyravian)* and sending the result, the userid, and the message authentication code to the server; *("The client sends id, auth_token, and protected_idpw_digest_new to the server" pg. 7, Peyravian)*
>
> wherein the server retrieves the new digest by exclusive-oring the mask with the received result ("To *retrieve idpw_digest_new, the server generates auth_token_mask ... and XORs it with the received protected_idpw_digest_new" pg.* 7, *Peyravian),* and wherein the server verifies the received message authentication code,
>
> and wherein if the received message authentication code is verified, the server changes the client password. ("If it is valid, the server sends a message to the client accepting the password change", pg. 7, Peyravian).
>
> It is inherent that if the password is changed, the old password will be replaced with a new password.
>
> Peyravian does not teach wherein the first set of values including a first random value, a large prime number, a primitive root of the large prime number, and a large random integer less than the large prime number minus one,
>
> or where the server generates the challenge token by exclusive-oring the first random value with a first hash, wherein the first hash is a hash of the following: a primitive root of a large prime number raised to a power, a digest of the client's userid and password, and a second random value.
>
> Peyvarian further does not teach wherein the client retrieves the server-generated random value from the challenge token and sends the server-generated random value and the userid to the server;
>
> wherein the server verifies the received server-generated random value from the client is correct by comparing the first random value received from the client with the server's stored value of the first random number.
>
> Trostle teaches a password based protocol using the Diffie-Hellman algorithm for key exchange. ("by using the Diffie-Hellman algorithm for key exchange and C's private

key only to sign this exchange, a much better level of security is achieved" Column 2, lines 23-25). Trostle teaches that "Diffie-Hellman key exchange, there are two publicly known numbers: a prime number p and an integer g that is a primitive root of p" (Column 4, lines 24-26). Trostle teaches generating a challenge by using a calculated shared key ($Dhkey_2 = X_2^{\wedge y2}$ mod p) to encrypt a hashed password digest (h(k(P')), a first random value s', and a random value c. Trostle further teaches that the first generated value s is returned and "upon receipt of message, C uses Dhkey to decrypt the message, obtaining, s' and $Y_3$ , C uses s' to authenticate S." (Column 6, lines 36-38). This authentication is done by comparing s' to the value that C originally generated and stored. *("C generates and stores random values c' and s"',Column 6, lines 14-15)*

It would have been obvious to one of ordinary skill in the art at the time of the invention to modify the password changing system in Peyravian with the Diffie-Hellman key based password system of Trostle.

The motivation is that Trostle teaches a known way of using the Diffie-Hellman algorithm for key exchange to change passwords. Trostle describes the motivation and steps for changing passwords in Column 6, lines 1-38.

The combined references of Peyravian and Trostle still do not explicitly teach sending a large prime number, a primitive root of the large prime number, and primitive root raised to a large random integer less than the large prime number minus one. The combined references also do not explicitly teach wherein the challenge token comprises a server generated random value XORed with a hash of the primitive root of the large prime number raised to a power, a digest of the client's userid and password, and the client generated random value, and then have the server generate a one-time authentication token and sends it to the client, giving it permission to access the server, wherein the client verifies the validity of the one-time authentication token.

It would have been obvious to one of ordinary skill in the art at the time of the invention to modify Trostle to first send a large prime number, a primitive root of the large prime number, and the primitive root raised to a large random integer, and to further include a challenge of a server generated random value XORed with a hash of the primitive root of the large prime number raised to a power, a digest of the client's userid and password, and the client generated random value and then have the server generate a one-time authentication token and sends it to the client, giving it permission to access the server, wherein the client verifies the validity of the one-time authentication token.

The motivation is a large prime number, a primitive root and a primitive root raised to a large number are all necessary for Diffie-Hellman key exchange, sending these values from a client to a server would have been within the skill of one in the art. The motivation to use a challenge token is that although Trostle uses the shared secret to encrypt, the password digest, first random value s' and the random value c, while the Applicant describes a challenge token that uses the random value s XORed with the hash of the shared secret, password digest and random value c, it would have been obvious to one of ordinary skill to generate a challenge using the challenge token for the following reasons.

In both cases the sender gives the receiver the necessary unencrypted value (Cid, [c, s', h(c'), h(k(P'))]$Dhkey_2$, $\underline{\mathbf{X}_2}$ Column 6, line 19 of Trostle, $g^y$ of the Application) to generate the shared key. Once the shared key (Dhkey) is generated, the receiver can then "unlock" the remaining values. In Trostle, the Dhkey, is used to decrypt the random values, where as in the instant application, a hash is calculated using the generated shared

key along with other values known by the client and then the hash is XORed with another random values used to authenticate.

As described in the previous office action the use of one-time pads and MACs are very well known in the art. (Searchsecurity.com and ATIS). The combination of Peravian and Trostle included each element as claimed (random value s', random value c, g^integer, password digest). The Examiner believes that one of ordinary skill in the art could have combined the elements as claimed by known methods, such as using a Message Authentication Code and a one-time pad. Furthermore one of ordinary skill in the art would have recognized that the results of the combination were predictable.

Finally, the Examiner interprets generating a one-time authentication token, wherein the client verifies the validity of the one-time token as repeating the steps of the challenge token and subsequent authentication. One of ordinary skill in the art would be able to duplicate the steps.

Regarding the computer program product and the method associated with the computer network above, the method is already included in the cited sections and it is inherent that a password changing system on a computer network requires a computer program product on a computer readable medium.

Final Office Action dated August 16, 2007, pages 4-9.

Amended claim 1, which is representative of the other rejected independent claims 9 and 17 with regard to similarly recited subject matter, reads as follows:

1. A computer network, comprising:

a client and a server connected by a network connection, wherein the client has a userid and a password associated with the client;

wherein the client requests access to the server by sending a first set of values to the server, wherein <u>the first set of values includes a client-generated random value, a large prime number, a primitive root of the large prime number, and the primitive root raised to a power of a large random integer less than the large prime number minus one;</u>

wherein <u>the server responds to the client by generating a one-time challenge token that depends at least on a server-generated random value and sending the challenge token to the client, wherein the server generates the challenge token by exclusive-oring the server-generated random value with a first hash, and wherein the first hash is a hash of the primitive root of the large prime number raised to a power, a digest of the client's userid and password, and the client-generated random value;</u>

wherein <u>the client retrieves the server-generated random value from the challenge token and sends the server-generated random value and the userid to the server;</u>

wherein <u>the server verifies the received server-generated random value from the client is correct by comparing the server-generated random value received from the client with the server's stored value of the server-generated random number, and if so, the server generates a one-time authentication token and sends it to the client, giving it permission to access the server;</u>

wherein <u>the client verifies the validity of the one-time authentication token received from the server;</u>

wherein if the client verifies that the one-time authentication token from the server is valid, the client changes the password by computing a hash of the userid and a new password to form a new digest, creating a mask, computing a message authentication code, and by exclusive-oring the mask with the new digest to form a result, and sending the result, the userid, and the message authentication code to the server;

wherein the server retrieves the new digest by exclusive-oring the mask with the received result, and wherein the server verifies the received message authentication code; and

wherein if the received message authentication code is verified, the server changes the client password by replacing a digest of at least the old password with a digest of at least the new password. (emphasis added)

The Examiner bears the burden of establishing a *prima facie* case of obviousness based on the prior art when rejecting claims under 35 U.S.C. § 103. *In re Fritch*, 972 F.2d 1260, 23 U.S.P.Q.2d 1780 (Fed. Cir. 1992). For an invention to be *prima facie* obvious, the prior art must teach or suggest all claim limitations. *In re Royka*, 490 F.2d 981, 180 USPQ 580 (CCPA 1974).

*Peyravian* and *Trostle*, taken alone or in combination, do not teach or suggest that "the client requests access to the server by sending a first set of values to the server, wherein the first set of values includes a client-generated random value, a large prime number, a primitive root of the large prime number, and the primitive root raised to a power of a large random integer less than the large prime number minus one; wherein the server responds to the client by generating a one-time challenge token that depends at least on a server-generated random value and sending the challenge token to the client, wherein the server generates the challenge token by exclusive-oring the server-generated random value with a first hash, and wherein the first hash is a hash of the primitive root of the large prime number raised to a power, a digest of the client's userid and password, and the client-generated random value; wherein the client retrieves the server-generated random value from the challenge token and sends the server-generated random value and the userid to the server; wherein the server verifies the received server-generated random value from the client is correct by comparing the server-generated random value received from the client with the server's stored value of the server-generated random number, and if so, the server generates a one-time authentication token and sends it to the client, giving it permission to access the server, wherein the client verifies the validity of the one-time authentication token received from the server," as recited in independent claims 1, 9, and 17.

*Peyravian* is directed to methods for protecting password transmission. The presently claimed invention refers to the cited *Peyravian* reference on page 3, of the specification and states that the schemes, such as in the cited *Peyravian* reference, do not provide protection against the offline password-guessing attack (i.e. dictionary attack) and denial of service attack. The claims of the present invention provide an improved method that provides this protection. In the cited *Peyravian* reference, the client only sends a userid and a client-generated random value when requesting access to the server and the server responds by sending a server-generated random value. The cited *Peyravian* reference does not provide the additional features and steps of claims 1, 9, and 17. *Peyravian* does not teach or suggest that "the client requests access to the server by sending a first set of values to the server, wherein the first set of values includes a client-generated random value, a large prime number, a primitive root of the large

prime number, and the primitive root raised to a power of a large random integer less than the large prime number minus one; wherein the server responds to the client by generating a one-time challenge token that depends at least on a server-generated random value and sending the challenge token to the client, wherein the server generates the challenge token by exclusive-oring the server-generated random value with a first hash, and wherein the first hash is a hash of the primitive root of the large prime number raised to a power, a digest of the client's userid and password, and the client-generated random value; wherein the client retrieves the server-generated random value from the challenge token and sends the server-generated random value and the userid to the server; wherein the server verifies the received server-generated random value from the client is correct by comparing the server-generated random value received from the client with the server's stored value of the server-generated random number, and if so, the server generates a one-time authentication token and sends it to the client, giving it permission to access the server, wherein the client verifies the validity of the one-time authentication token received from the server," as recited in independent claims 1, 9, and 17.

*Trostle* is directed to a method for a first participant to establish a shared secret with a second participant, where the first participant and the second participant share a password-based first master key and a hash function includes sending a first message including a first private value for the second participant and a first authenticator for the second participant encrypted with the first master key. The first message also includes a first hashed authenticator for the first participant encrypted with a first shared secret key. The first message also includes a first public value for the first participant. The first participant receives a second message, the second message including the first authenticator for the second participant and a first public value for the second participant encrypted with the first shared secret key. The first participant sends a third message, the third message including the first authenticator for the first participant, a second hashed authenticator for the first participant, a second authenticator for the second participant and a second master key encrypted with a second shared secret key. The third message also includes a second public value for the first participant. A fourth message is received by the first participant, the fourth message including a second authenticator for the second participant and a second public value for the second participant encrypted with the second shared secret key. *Trostle* does not teach or suggest that "the client requests access to the server by sending a first set of values to the server, wherein the first set of values includes a client-generated random value, a large prime number, a primitive root of the large prime number, and the primitive root raised to a power of a large random integer less than the large prime number minus one; wherein the server responds to the client by generating a one-time challenge token that depends at least on a server-generated random value and sending the challenge token to the client, wherein the server generates the challenge token by exclusive-oring the server-generated random value with a first hash, and wherein the first hash is a hash of the primitive root of the

large prime number raised to a power, a digest of the client's userid and password, and the client-generated random value; wherein the client retrieves the server-generated random value from the challenge token and sends the server-generated random value and the userid to the server; wherein the server verifies the received server-generated random value from the client is correct by comparing the server-generated random value received from the client with the server's stored value of the server-generated random number, and if so, the server generates a one-time authentication token and sends it to the client, giving it permission to access the server, wherein the client verifies the validity of the one-time authentication token received from the server," as recited in independent claims 1, 9, and 17.

*Trostle* teaches a different password based protocol for secure communications between two parties who share a relatively insecure secret. In *Trostle*, prior to beginning a message exchange the two parties share knowledge of the secret password, a one-way hash function, an encryption key, another key, a publicly known prime number, and a publicly known primitive root of the prime number. The message exchange is separated into three stages: the "Setup Initial State" stage, the "Generate New Password" stage, and the "Normal Exchange" stage. The "Generate New Password" stage referred to in the rejection of claims 1, 9, and 17 is described in *Trostle* as follows:

> Generate New Password
>
> The initial exchange of messages **104** and **106** is vulnerable to a password chaining attack. This is because if an attacker correctly guesses K, he can obtain $y_1$ and use both $y_1$ and $X_1$ to generate $DHkey_1$. The attacker may then compare s in message **104** with s in message **106**. If the two values are the same, the attacker knows he has obtained the correct $DHkey_1$ and can therefore decrypt subsequent messages. Therefore, according to the present invention, the master password is changed. If the principle is a user, the user is prompted to enter a new password (P'). If the principle is a process, the process generates its own new password. When C **100** has P', it computes and stores a new master key h(k(P')). Additionally, C **100** generates and stores random values c' and s' and Diffie-Hellman pair ($x_2$, $X_2$). C **100** also calculates $DHkey_2 = X_2^{y2} \bmod p$. C **100** then sends
>
> Cid, [c, s', h(c'), h(k(P'))]$DHkey_2$, $X_2$ (msg.**108**)
>
> to S **102** at reference numeral **108**. In message **108**, the value c is used to authenticate message **108** to S **102** and the value h(c') is used by C **100** to authenticate a future message.
>
> Next, S **102** uses the value of Y.sub.2 stored earlier and the value of X.sub.2 sent in message 108 to calculate $DHkey_2 = X_2^{y2} \bmod p$. S **102** uses DHkey.sub.2 to decrypt message **108**, obtaining c, h(c') and h(k(P')). S **102** generates Diffie-Hellman pair (y.sub.3, Y.sub.3). S **102** stores y.sub.3, $Y_3$ and h(k(P')). S **102** authenticates C **100** by computing h(c) and comparing it to the previous value of h(c). If the two values are the same, S **102** knows that C **100** sent the message. At reference numeral **110**, S **102** sends
>
> Sid, [s', $Y_3$]$DHkey_2$ (msg.**110**)
>
> to C **100**.
>
> Upon receipt of message **110**, C **100** uses $Dhkey_2$ to decrypt the message, obtaining s' and $Y_3$. C **100** uses s' to authenticate S. C **100** stores $Y_3$.

*Trostle*, column 6, lines 1-38.

*Trostle* teaches a different method and way of using Diffie-Hellman key agreement scheme. The cited portion of *Trostle* teaches using the Diffie-Hellman key to encrypt: a hash value of an encrypted new password, a hash value of a client-generated random value c', a client-generated random value s', and a client-generated random value c. Claims 1, 9, and 17 of the present invention recite that the server generates the challenge token by exclusive-oring a server-generated random value with a first hash. This first hash is a hash of the primitive root of the large prime number raised to a power, a digest of the client's userid and password, and a client-generated random value. This digest of the client's userid and password is for the current/old password. In the claims of the present invention, the password is not changed until the client verifies a separate one-time authentication token from the server. The Examiner alleges that it would be obvious to one skilled in the art to combine the elements of known methods to achieve the claimed invention. Applicants respectfully disagree. In addition, the present invention avoids the need for public-key or symmetric-key cryptosystems. In the present invention, the Diffie-Hellman key agreement scheme is used by the client and server to establish a shared secret to protect exchanges; and the method of the present invention does <u>not</u> use static or established Diffie-Hellman ephemerals between the client and sever. To the contrary, *Trostle's* method uses two publicly known numbers: a prime number p and an integer g that is a primitive root of p.

*Peyravian* and *Trostle* fail to teach or suggest that "the client requests access to the server by sending a first set of values to the server, wherein the first set of values includes a client-generated random value, a large prime number, a primitive root of the large prime number, and the primitive root raised to a power of a large random integer less than the large prime number minus one; wherein the server responds to the client by generating a one-time challenge token that depends at least on a server-generated random value and sending the challenge token to the client, wherein the server generates the challenge token by exclusive-oring the server-generated random value with a first hash, and wherein the first hash is a hash of the primitive root of the large prime number raised to a power, a digest of the client's userid and password, and the client-generated random value; wherein the client retrieves the server-generated random value from the challenge token and sends the server-generated random value and the userid to the server; wherein the server verifies the received server-generated random value from the client is correct by comparing the server-generated random value received from the client with the server's stored value of the server-generated random number, and if so, the server generates a one-time authentication token and sends it to the client, giving it permission to access the server, wherein the client verifies the validity of the one-time authentication token received from the server," as recited in independent claims 1, 9, and 17. Therefore, the alleged combination of *Peyravian* and *Trostle* does not teach or suggest these features. In addition, the Final Office Action states that the "combined references of *Peyravian* and *Trostle* still do not explicitly teach sending a large prime number, a primitive root of the

large prime number, and primitive root raised to a large random integer less than the large prime number minus one. The combined references also do not explicitly teach wherein the challenge token comprises a server generated random value XORed with a hash of the primitive root of the large prime number raised to a power, a digest of the client's userid and password, and the client generated random value, and then have the server generate a one-time authentication token and sends it to the client, giving it permission to access the server, wherein the client verifies the validity of the one-time authentication token." Accordingly, Applicants respectfully request withdrawal of the rejection of claims 1, 9, and 17 under 35 U.S.C. § 103(a).

## III. Conclusion

It is respectfully urged that the subject application is patentable over the cited references and is now in condition for allowance.

The Examiner is invited to call the undersigned at the below-listed telephone number if in the opinion of the Examiner such a telephone conference would expedite or aid the prosecution and examination of this application.

DATE: November 12, 2007

Respectfully submitted,

/Gerald H. Glanzman/

Gerald H. Glanzman
Reg. No. 25,035
Yee & Associates, P.C.
P.O. Box 802333
Dallas, TX 75380
(972) 385-8777
Attorney for Applicants

GHG/VJA